A solid understanding of HTML structure is fundamental for creating well-organized, styled, and accessible web pages. This question tests your grasp of how elements are uniquely identified or grouped in the DOM for styling or scripting.

Answer:
`id` is a unique identifier for a single element on the page. It must be used once per page and is often used for JavaScript targeting or CSS styling.
`class` is reusable and can be applied to multiple elements, making it perfect for grouping similar styles or behavior.

```
<div id="header"></div>
<div class="card"></div>
<div class="card"></div>
```

---

2. 📏 What's the difference between `relative`, `absolute`, and `fixed` positioning in CSS?

Understanding CSS positioning is key to building fluid layouts. Knowing when and how elements shift on the page, whether in relation to their parent, the document, or the viewport, can help you construct flexible designs.

Answer:

- `relative`: Moves an element relative to its normal position without affecting the layout of others.
- `absolute`: Positions the element based on the nearest positioned ancestor (non-static).
- `fixed`: Positions the element relative to the viewport — it stays in place during scrolling.

---

3. 🧠 What is the DOM?

The DOM is the bridge between your HTML and JavaScript. Mastering DOM manipulation enables dynamic interactivity, like toggling modals, updating text, or creating responsive interfaces without reloading the page.

Answer:
The **Document Object Model** is a tree-like structure representing HTML elements. JavaScript uses it to modify structure, style, and content on the fly.

Example: `document.getElementById('myDiv').innerText = "Hello";`

---

4. ➡ What are arrow functions in JavaScript?

Arrow functions not only simplify syntax but also change how the `this` keyword behaves. This is especially important when dealing with methods inside classes or asynchronous callbacks, where traditional function expressions can lead to unexpected `this` references.

Answer:
Arrow functions are a concise way to write functions. Unlike regular functions, they **do not bind their own `this`**, making them ideal for callbacks.

```
const greet = name => `Hello, ${name}`;
```

---

5. 🔍 What's the difference between `==` and `===`?

JavaScript can be confusing due to implicit type conversion. Using `===` (strict equality) reduces bugs by ensuring both value and type match, a best practice in modern JavaScript.

Answer:

- `==` compares value, allowing type coercion.
- `===` compares both value and type, making it safer and more predictable.

Example: `5 == "5"` → `true`, but `5 === "5"` → `false`.

Final Round AI
Fast Track Your Career
With Final Round AI, your path to a job offer is faster than ever.
From personalized coaching to AI-powered tools, we make job
searching smarter and more efficient.

Try Final Round AI for FREE today! 🔥

6. 🧵 What is event delegation in JavaScript?

In dynamic apps where elements are added or removed frequently, attaching event listeners to each new element can become inefficient. Event delegation improves performance by utilizing event bubbling to catch events higher in the DOM.

Answer:
Instead of adding event listeners to each child element, we attach a single listener to a parent and catch events as they bubble up. It reduces memory usage and simplifies code.

---

7. ⏱️ What is the difference between synchronous and asynchronous code?

Modern applications rely on asynchronous code to remain responsive. Understanding this difference helps you build apps that can fetch data or process tasks in the background without freezing the interface.

Answer:

- **Synchronous** code blocks execution until complete.
- **Asynchronous** code (e.g., with Promises or async/await) allows other tasks to continue while waiting for operations like API responses.

---

8. 🎣 What are React Hooks?

Hooks allow you to reuse stateful logic across components and write cleaner, more modular code. They're a cornerstone of functional component design in modern React applications.

Answer:
Hooks are functions like `useState`, `useEffect`, and `useRef` that let you manage state, lifecycle, and refs in functional components, replacing class components in many cases.

---

9. ⚖️ What is the difference between `props` and `state` in React?

Props and state serve different roles in React. Props let parent components pass data down, while state enables a component to manage its internal data and re-render on change. Understanding both is key to React's data flow model.

Answer:

- `props`: Immutable data passed from parent to child components.
- `state`: Mutable local data managed within the component.

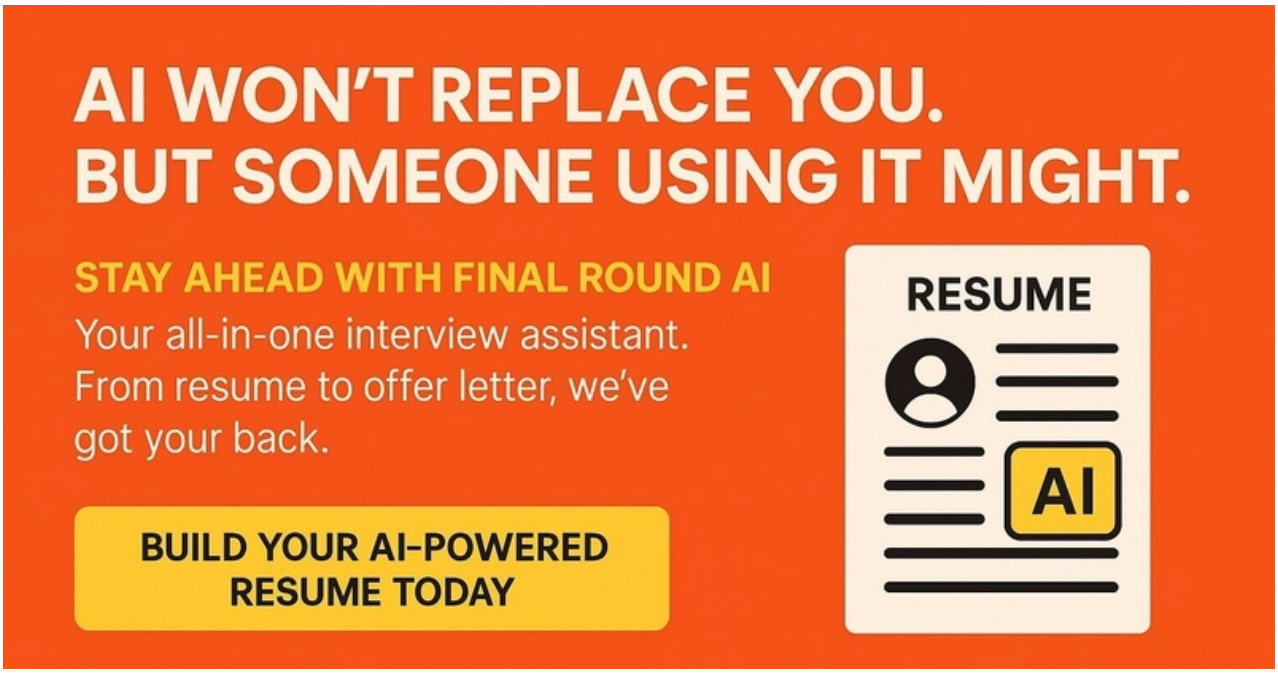
```
function Welcome({ name }) {
  const [count, setCount] = useState(0);
  return <h1>Hello {name}, you clicked {count} times</h1>;
}
```

---

10. 💭 What is the Virtual DOM?

React uses the Virtual DOM to boost performance by minimizing direct DOM manipulation. Changes are first calculated virtually, and only the necessary real DOM updates are applied, leading to faster and more efficient rendering.

Answer:
The Virtual DOM is a lightweight JS representation of the actual DOM. React updates the Virtual DOM first, then computes the difference (diffing) and updates the real DOM efficiently.



11. 🎛️ What is a controlled component in React?

Controlled components ensure form inputs reflect the component's state. This synchronization makes it easier to validate, manipulate, or submit data programmatically, offering more control over UI behavior.

Answer:
A controlled component has its value managed by React via `state`. The form's behavior is predictable and easy to debug.

```
<input value={email} onChange={e => setEmail(e.target.value)} />
```

---

12. 🔄 Explain lifecycle methods in React.

Lifecycle methods help manage a component's creation, update, and removal. Hooks like useEffect simplify this process in functional components, enabling side effects like API calls or subscriptions with clean-up logic.

Answer:
Functional components use hooks like:

> `useEffect`: replaces `componentDidMount`, `componentDidUpdate`, and `componentWillUnmount`.

Class-based examples:

- `componentDidMount()`: runs after component mounts.
- `componentWillUnmount()`: cleans up.

---

13. 📱 What is responsive design?

Responsive design ensures your website adapts seamlessly to different screen sizes. By using fluid layouts and media queries, you enhance usability and accessibility across phones, tablets, and desktops.

Answer:
Responsive design uses flexible grids, media queries, and flexible images to make websites look great on any device.

```
@media (max-width: 768px) {
  .container { flex-direction: column; }
}
```

---

14. 🧪 What is unit testing in frontend development?

Unit tests help catch bugs early by verifying that each component or function behaves as expected. Writing tests builds confidence in your codebase, reduces regressions, and facilitates safe refactoring.

Answer:
Unit testing checks individual functions or components. Tools: **Jest**, **React Testing Library**, **Mocha**.

---

15. 💤 What is lazy loading?

Lazy loading delays the loading of components or resources until they are needed. This improves initial page load times and overall performance, especially for larger applications with many routes or components.

Answer:
React supports lazy loading components using `React.lazy()` and `Suspense`.

```
const LazyComponent = React.lazy(() => import('./MyComponent'));
```

Try Final Round AI for FREE today! 🔥

16. 🚧 What is Webpack?

Webpack bundles all your assets, avaScript, CSS, images, into optimized files. This makes your app faster and more efficient, while also supporting modern features like code splitting and hot module replacement.

Answer:
Webpack is a module bundler. It compiles JS, CSS, images, and other assets into optimized bundles.

---

17. 🔐 What is XSS and how to prevent it?

XSS allows attackers to inject scripts into your pages. Preventing XSS is crucial and involves escaping outputs, sanitizing inputs, and relying on frameworks like React that automatically mitigate many common vulnerabilities.

Answer:
XSS (Cross-Site Scripting) is when malicious JS is injected into a page. Prevent it by:

- Escaping output
- Using React (which auto-escapes JSX)
- Sanitizing user input

---

18. 🎨 What is CSS specificity?

Specificity determines which CSS rules apply when multiple rules target the same element. Understanding this helps you write cleaner, conflict-free styles and avoid relying on `!important` to override unwanted behavior.

Answer:
Specificity determines which CSS rule takes precedence. Inline > IDs > Classes > Elements.

---

19. 🎭 What is the difference between `null`, `undefined`, and `NaN` in JavaScript?

These three values represent different types of "absence" or "invalid" data in JavaScript. Misunderstanding them can lead to logic errors and unexpected behavior in your applications.

Answer:

- `undefined`: variable declared but not assigned.
- `null`: intentionally empty.
- `NaN`: result of an invalid numeric operation.

---

20. 🧱 What are the building blocks of a modern frontend stack?

Modern frontend development goes beyond writing HTML. A full stack includes component-based frameworks, state management, API calls, testing tools, and bundlers, all working together to create efficient UIs.

Answer:
A typical frontend stack includes:

- **HTML/CSS/JS**
- **React/Vue/Angular**
- **Webpack/Vite**
- **Axios/Fetch**
- **Testing tools like Jest**

NEVER FREEZE UP IN AN INTERVIEW AGAIN
Final Round AI's Interview Copilot gives you real-time suggestions during interviews.
TRY INTERVIEW COPILOT FOR FREE TODAY!

🔧 20 Backend Interview Questions

> On the backend side, recruiters focus on your ability to design robust APIs, manage databases efficiently, and handle authentication, performance, and security concerns. These questions explore key backend skills from RESTful API design and middleware to database optimization, caching, and server-side architecture.

21. 🌍 What is REST API?

RESTful APIs standardize communication between client and server. They make backend services predictable and scalable, leveraging the existing web infrastructure (like HTTP methods) to create, read, update, and delete resources.

Answer:
REST (Representational State Transfer) uses HTTP methods (GET, POST, PUT, DELETE) to operate on resources via stateless APIs.

---

22. 🚧 What is middleware in Express.js?

Middleware functions allow you to modularize your Express app. They can preprocess requests, handle errors, check authentication, and perform logging, making your backend more maintainable and reusable.

Answer:
Middleware functions have access to `req`, `res`, and `next`. They're used for logging, auth, validation, etc.

```
app.use((req, res, next) => {
  console.log('Request received');
  next();
});
```

---

23. 🔐 What is JWT and how does it work?

JWTs are compact, self-contained tokens that verify identity without server-side sessions. They enable scalable stateless authentication and are commonly used in APIs to protect routes and resources.

Answer:
JWT (JSON Web Token) encodes user data into a signed token. It's stored on the client and sent with each request in headers.

---

24. 🤔 Difference between PUT and PATCH?

Using the right HTTP method communicates intent clearly. PUT is ideal for replacing a resource entirely, while PATCH is better for partial updates, helping APIs remain intuitive and flexible.

Answer:

- `PUT`: replaces the entire resource.
- `PATCH`: updates specific fields.

---

25. 🌌 What is the event loop in Node.js?

The event loop is what allows Node.js to handle many requests at once without multithreading. It processes asynchronous callbacks (like I/O or timers) efficiently, enabling fast, non-blocking backend services.

Answer:
The event loop allows Node.js to handle async operations (I/O) without blocking the main thread.

Final Round AI
Real-Time AI Support
for Your Tech Interview Success
More than just preparation – Interview Copilot actively supports you during your live interview, ensuring you never feel stuck.
Get hired at top tech companies!

26. 📦 What is npm and why use it?

npm is more than just a package manager, it's a vast ecosystem of open-source tools. It simplifies dependency management, version control, and script execution in Node.js applications.

Answer:
npm (Node Package Manager) manages libraries and dependencies in Node.js projects.

---

27. 🧮 What are environment variables?

Environment variables keep sensitive information like API keys out of your source code. They support environment-specific configurations and are critical for secure and portable applications.

Answer:
Stored in `.env` files and accessed via `process.env`, they help keep config secure and flexible.

---

28. 🧪 What is unit testing in backend?

Unit tests ensure backend logic, services, and endpoints work as intended. They improve reliability, document behavior, and speed up debugging when something breaks in production.

Answer:
Use tools like **Jest**, **Mocha**, or **Pytest** to test routes, logic, and services in isolation.

---

29. 🗂️ What is MVC architecture?

MVC organizes backend code by separating responsibilities. Models handle data, Views represent output (often JSON in APIs), and Controllers manage request-response logic, making large apps easier to scale and maintain.

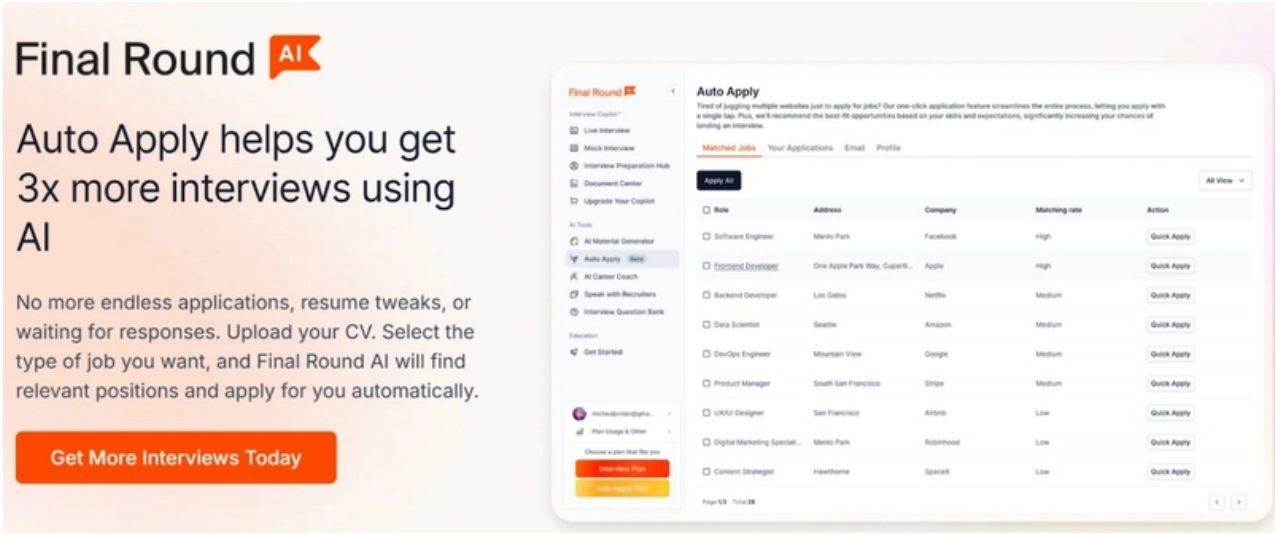
Answer:
MVC (Model-View-Controller) separates data, UI, and control logic. It improves code organization and maintenance.

---

30. 🐘 Why use PostgreSQL over other DBs?

PostgreSQL is a powerful, open-source relational database with strong standards compliance. It supports advanced features like JSONB, full-text search, and complex queries, making it a versatile choice for many applications.

Answer:
PostgreSQL offers strong ACID compliance, supports JSON, indexing, and is ideal for relational data.


Final Round AI
Auto Apply helps you get 3x more interviews using AI
No more endless applications, resume tweaks, or waiting for responses. Upload your CV. Select the type of job you want, and Final Round AI will find relevant positions and apply for you automatically.
Get More Interviews Today
Auto Apply

31. ⚖️ What is the difference between SQL and NoSQL databases?

The choice between SQL and NoSQL affects your app's scalability and data integrity. SQL enforces schema and relationships, while NoSQL offers flexibility and speed for unstructured or rapidly changing data.

Answer:

- **SQL (Relational):** Structured schema, supports JOINs, ACID-compliant (e.g., PostgreSQL, MySQL).
- **NoSQL (Non-relational):** Schema-less, scalable horizontally, great for unstructured data (e.g., MongoDB, Firebase).

---

32. ⚙️ What is ORM, and why would you use it?

ORMs abstract away SQL syntax and let you interact with your database using objects. This improves developer productivity, maintains consistency, and reduces repetitive boilerplate code.

Answer:
ORM (Object Relational Mapping) tools like Sequelize (Node.js) or Django ORM (Python) allow you to interact with databases using object-oriented code instead of raw SQL.

```
# Django example
User.objects.filter(email='test@example.com')
```

---

33. 🧵 What is multithreading, and how is it handled in backend systems?

Multithreading boosts performance by running tasks in parallel. While Node.js uses a single thread with asynchronous callbacks, languages like Java and Python can manage multiple threads for intensive processing.

Answer:
Multithreading allows concurrent execution of tasks. In Node.js, it's single-threaded with an event loop, while Python or Java can use real threads (via `threading`, `concurrent.futures`, etc.).

---

34. 🏁 What is rate limiting, and how can you implement it?

Rate limiting protects your backend from abuse, denial-of-service attacks, or accidental overloads. It ensures fair usage and can be implemented using middleware, reverse proxies, or third-party services.

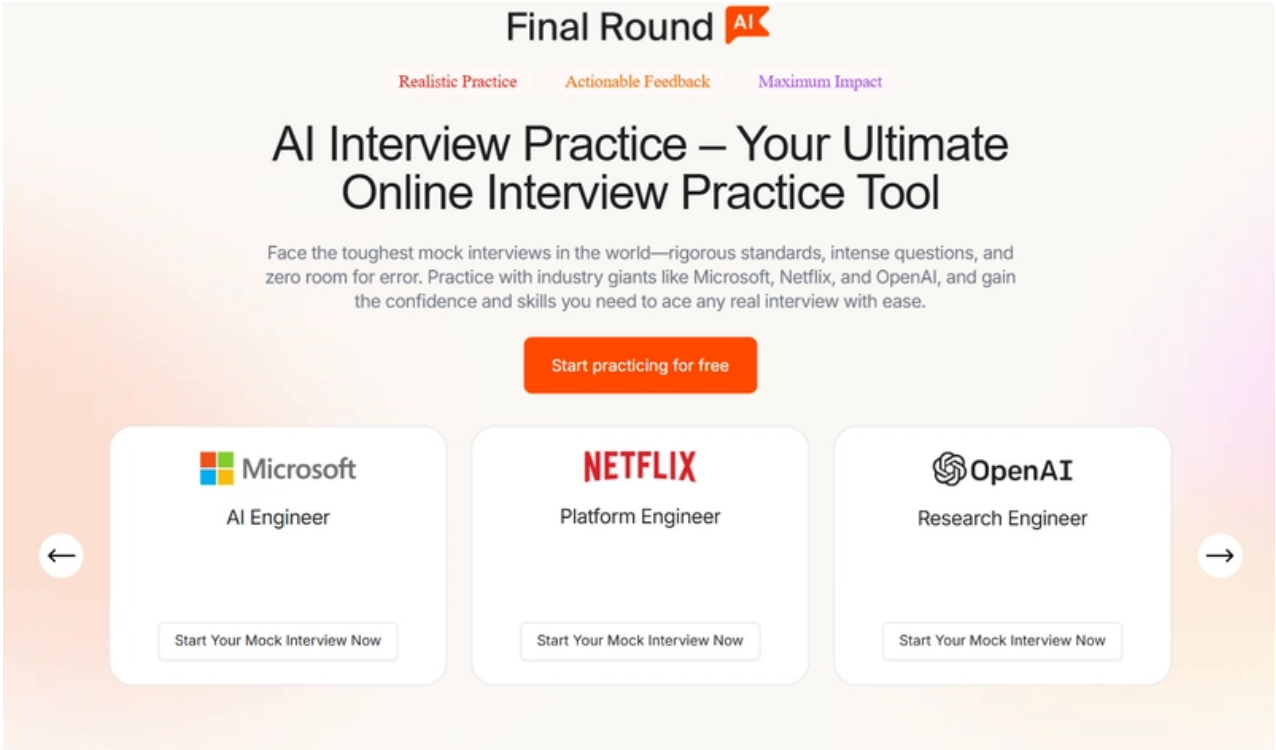
Answer:
Rate limiting restricts how many requests a client can make within a time window. You can use packages like `express-rate-limit` in Node.js or `throttling` in Django REST Framework.

---

35. 🛡️ What is CORS and why is it important?

CORS controls which origins can access your API. It’s vital for browser security and must be configured correctly to allow safe and intended communication between frontend apps and your backend server.

Answer:
CORS (Cross-Origin Resource Sharing) is a browser policy that blocks unknown domains from accessing your APIs unless explicitly allowed. You can configure headers like `Access-Control-Allow-Origin` to control access.


Final Round AI
Realistic Practice
Actionable Feedback
Maximum Impact
AI Interview Practice – Your Ultimate Online Interview Practice Tool
Face the toughest mock interviews in the world—rigorous standards, intense questions, and zero room for error. Practice with industry giants like Microsoft, Netflix, and OpenAI, and gain the confidence and skills you need to ace any real interview with ease.
Start practicing for free
Microsoft
AI Engineer
Start Your Mock Interview Now
NETFLIX
Platform Engineer
Start Your Mock Interview Now
OpenAI
Research Engineer
Start Your Mock Interview Now

36. 💣 What causes memory leaks in backend applications?

Memory leaks degrade performance and can crash servers over time. They occur when allocated memory is no longer needed but isn't released, slowly consuming resources until the system becomes sluggish or fails entirely. In backend systems that run continuously, even small leaks can accumulate and lead to major downtime or crashes if not detected early.

Answer:
Common causes include:

- Global variables not being cleared
- Forgotten timers or listeners
- Closures holding references Use profiling tools (e.g., Node's `--inspect`, memory profilers) to identify leaks.

---

37. 📊 How do you monitor and log backend applications?

Proper observability allows teams to detect anomalies, measure system health, and respond to incidents quickly. Logging captures key events and errors, while monitoring tools provide metrics and alerting, both of which are critical for maintaining uptime and performance in real-world environments.

Answer:
Use tools like:

- **Winston**, **Morgan** (Node.js logging)
- **Sentry**, **Datadog**, **Prometheus** for metrics and errors
- Store logs to external systems like ELK (Elasticsearch + Logstash + Kibana)

---

38. ⚡ What is GraphQL, and how is it different from REST?

Modern APIs are moving toward GraphQL for its flexibility and performance. It gives clients more control over the data they retrieve, improving performance and reducing bandwidth usage. Unlike REST, which can require multiple requests for related data, GraphQL enables precise queries that return all needed data in a single request, making it ideal for complex frontends like SPAs and mobile apps.

Answer:
GraphQL is a query language for APIs where clients specify exactly what data they need.

- Pros: One endpoint, no over-fetching
- REST: Multiple endpoints, predefined responses Example:

```
query {
  user(id: 1) {
    name
    email
  }
}
```

⛶⛶⛶⛶

---

39. 🧬 What are microservices, and when would you use them?

Microservices enable scalable and modular backend systems. They allow development teams to build applications as a collection of smaller, independently deployable services. Each service can evolve separately, making it easier to scale specific components, update features without downtime, and assign ownership of services to different teams.

Answer:
Microservices are small, independent services that communicate via APIs. You use them when:

- You want independent deployment
- Teams are working on different modules
- The app needs to scale horizontally

---

40. 🏗️ What's the difference between monolithic and microservice architecture?

Monolithic applications are simpler to develop and deploy initially but can become rigid and hard to manage as complexity grows. Microservice architectures introduce more complexity upfront but provide greater flexibility, fault isolation, and team autonomy as the system scales. Choosing between them depends on project size, team capacity, and long-term goals.

Answer:

- **Monolith:** One large codebase, tightly coupled. Easier to start, harder to scale.
- **Microservices:** Decoupled services. Harder to build initially, easier to maintain long-term with large teams and complex apps.

🎉 Final Thoughts

Preparing for fullstack interviews in 2025 means **understanding both the frontend and backend deeply**, from DOM manipulation to async JavaScript, from REST APIs to database architecture.

Don't just memorize, try building a small project that implements many of these patterns and concepts. 🔥